Counting permutation patterns and calculating iterated sums

J. **Diehl** (University of Greifswald)

Berlin 04.02.2026

Algebraic and geometric aspects of signatures and rough analysis

Overview

Counting permutation patterns

Counting chirotope patterns

Fast iterated sums for tensor-to-tensor layers

Λ : *"large" permutation*

σ : *"small" permutation.*

For a subset $I \subset [|\Lambda|]$ consider the **standardized** subsequence at these indices. For example

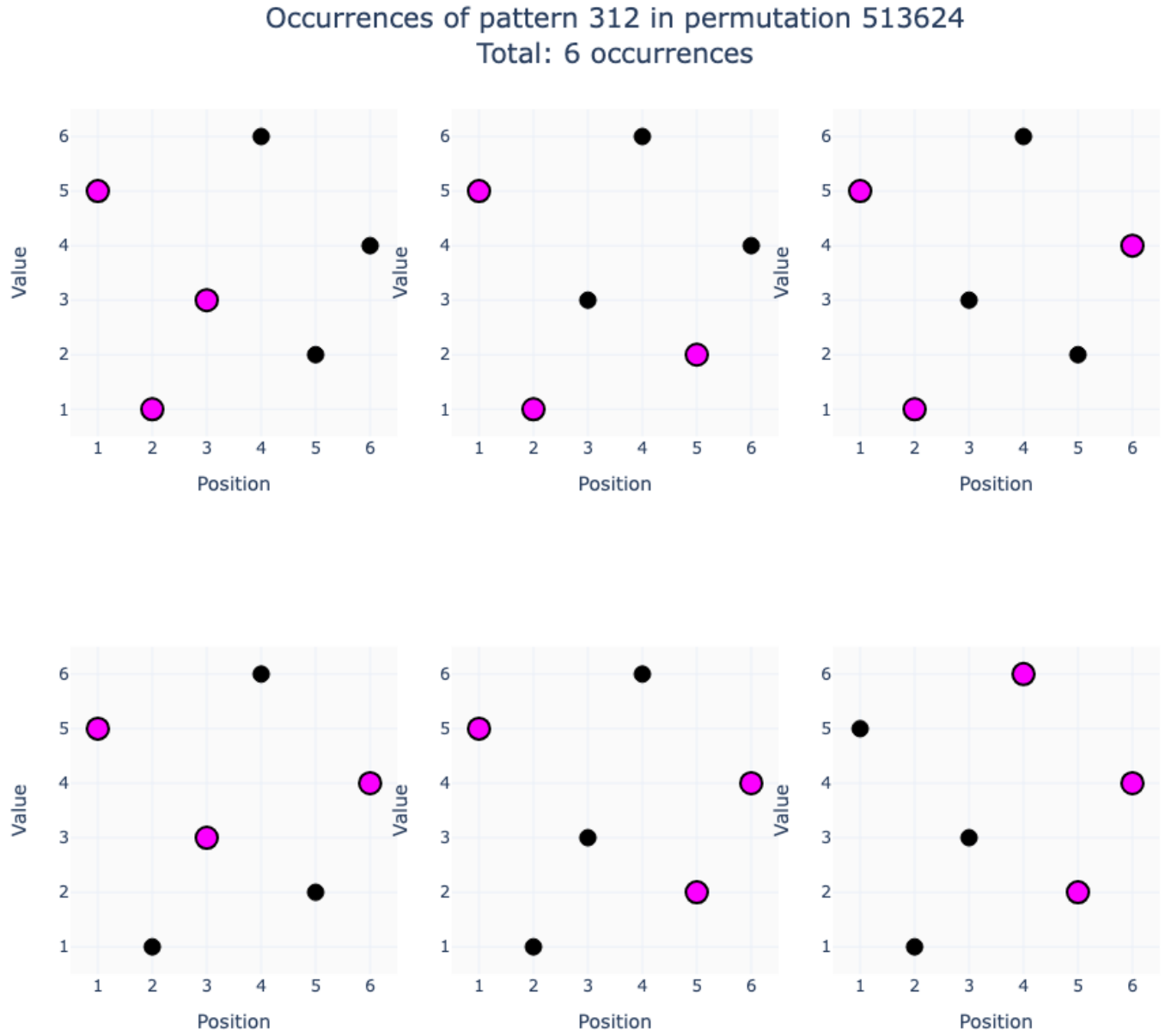
$$\mathrm{Std}([5\,1\,3\,6\,2\,4]|_{\{1,2,6\}}) = [3\,1\,2]$$

The number of times the small appears in the large one:

$$\begin{aligned}\#_\sigma(\Lambda) &:= \#\{\text{"copies" of } \sigma \text{ in } \Lambda\} \\ &:= \#\{I \subset [|\Lambda|] \;\mid\; \mathrm{Std}(\Lambda|_I) = \sigma\}.\end{aligned}$$

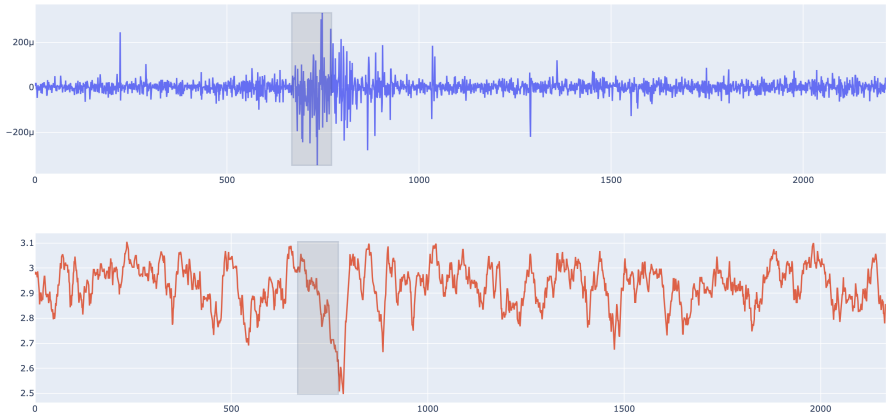
For example

$$\#_{312}([5\,1\,3\,6\,2\,4]) = 6.$$


Occurrences of pattern 312 in permutation 513624
Total: 6 occurrences
Value
Position
1 2 3 4 5 6

Permutation pattern counting appears in:

- Nonparametric independence tests (e.g. Hoeffding, Bergsma-Dassios-Yanagimoto).
- Permutation entropy of time series.



- The concept of σ-avoiding permutations.
  For example (Knuth): Λ can be stack-sorted iff $\#_{[1\,3\,2]}(\Lambda) = 0$.
- Their limit as $|\Lambda| \to \infty$ (permutons).

 The naive algorithm has time-complexity $\mathcal{O}(|\Lambda|^{|\sigma|})$. 

C. Even-Zohar and C. Leng. "Counting small permutation patterns". In: ACM-SIAM SODA. 2021:

$\rightsquigarrow$ linear complexity for a subspace of patterns.

The crucial idea: count subpatterns "without memory".

What memory? or **Why is pattern counting hard (on paper)?**

Consider again $\#_{312}$. We need to find three points $i_1, i_2, i_3 \in [|\Lambda|]$ such that

$$i_1 < i_2 < i_3$$
$$\Lambda_{i_1} > \Lambda_{i_3} > \Lambda_{i_2}.$$

So,

- i_1 needs to "know" about i_3 and i_2,
- i_2 needs to "know" about i_1 and i_2,
- i_3 needs to "know" about i_1 and i_2.

This makes (efficient) calculation difficult.

C. Even-Zohar and C. Leng. "Counting small permutation patterns". In: ACM-SIAM SODA. 2021:

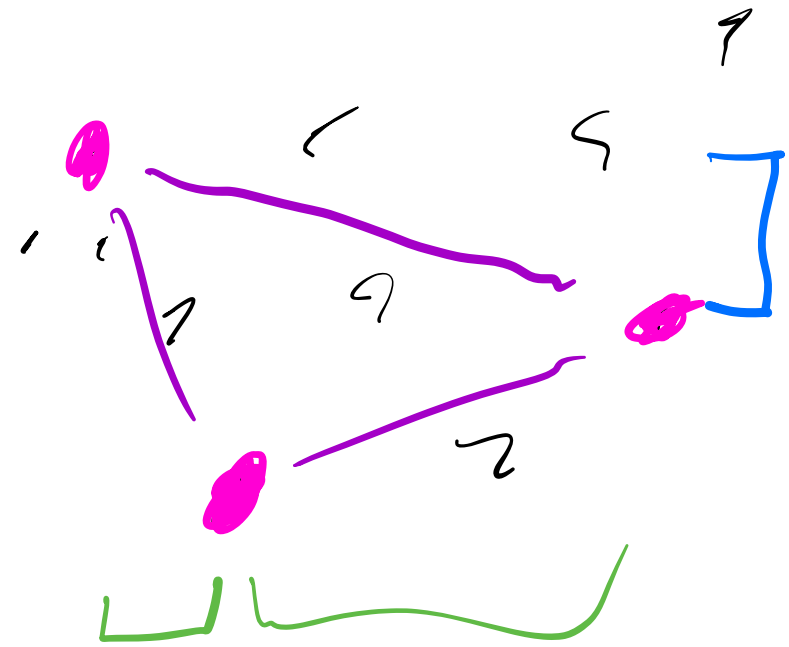
$\rightsquigarrow$ linear complexity for a subspace of patterns.

The crucial idea: count subpatterns "without memory".

What memory? or **Why is pattern counting hard (on paper)?**

Consider again $\#_{312}$. We need to find three points $i_1, i_2, i_3 \in [|\Lambda|]$ such that

$$i_1 < i_2 < i_3$$
$$\Lambda_{i_1} > \Lambda_{i_3} > \Lambda_{i_2}.$$



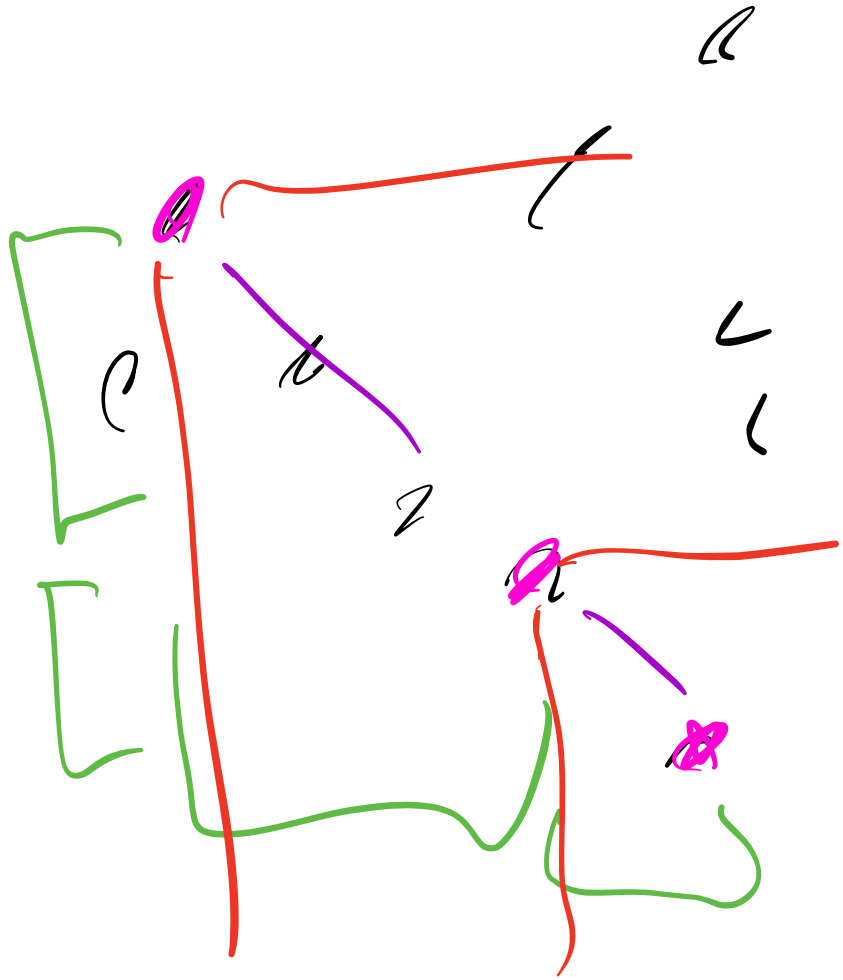
So,

- i_1 needs to "know" about i_3 and i_2,
- i_2 needs to "know" about i_1 and i_2,
- i_3 needs to "know" about i_1 and i_2.

This makes (efficient) calculation difficult.

Consider instead the pattern $\#_{321}$. We need to find three points $i_1, i_2, i_3 \in [|\Lambda|]$ such that

$$i_1 < i_2 < i_3$$
$$\Lambda_{i_1} > \Lambda_{i_2} > \Lambda_{i_3}.$$

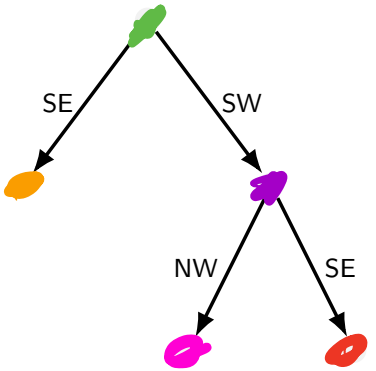


Here,

- i_1 only needs to "know" about i_2,
- i_2 needs to "know" about i_1, i_3,
- i_3 only needs to "know" about i_2.

corner tree

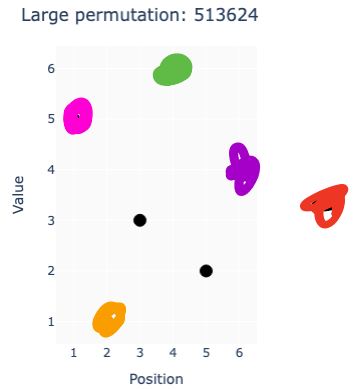
$\mathcal{T} = (V(\mathcal{T}), E(\mathcal{T}), \mathfrak{e} : E(\mathcal{T}) \to \{\mathrm{NE}, \mathrm{SE}, \mathrm{SW}, \mathrm{NW}\})$.



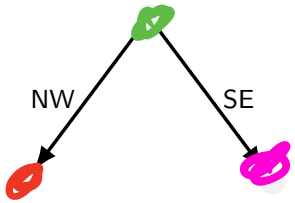
corner tree occurences

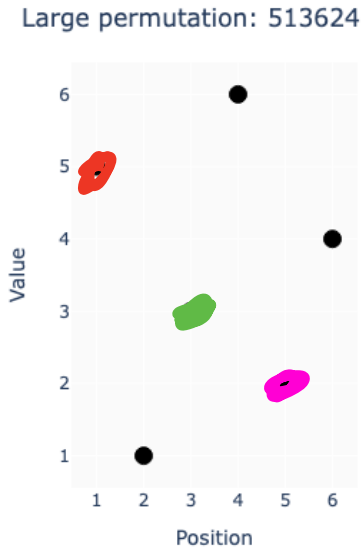
$\#_{\mathcal{T}}(\Lambda) := \#\{f : V(\mathcal{T}) \to [|\Lambda|]$ "compatible with the $\mathfrak{e}$"$\}$.

condition \ $\mathfrak{e}(v, v')$	NE	NW	SE	SW
$f(v') < f(v)$	×	✓	×	✓
$f(v') > f(v)$	✓	×	✓	×
$\Lambda(f(v')) < \Lambda(f(v))$	×	×	✓	✓
$\Lambda(f(v')) > \Lambda(f(v))$	✓	✓	×	×



One more example:





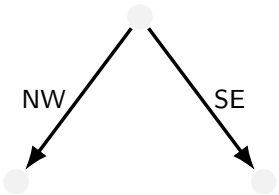
In general

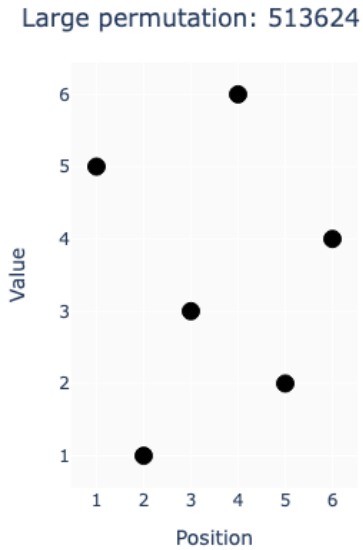
$$\#\tau = \sum_{\sigma} c_\sigma(\mathcal{T}) \#\sigma,$$

for some coefficients $c_\sigma(\mathcal{T}) \in \mathbb{N}$.[1]

[1] C. Even-Zohar and C. Leng. "Counting small permutation patterns". In: ACM-SIAM SODA. 2021.

One more example:





In general

$$\#\tau = \sum_{\sigma} c_\sigma(\mathcal{T}) \#\sigma,$$

for some coefficients $c_\sigma(\mathcal{T}) \in \mathbb{N}$.[1]

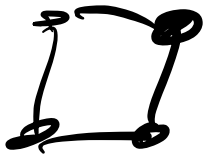
[1] C. Even-Zohar and C. Leng. "Counting small permutation patterns". In: ACM-SIAM SODA. 2021.

Theorem (Even-Zohar/Leng '21)

- *Up to level* 3 *one gets* all *permutation patterns via corner trees.*
- *Up to level* 4 *one gets all, except for a subspace of dimension one.*
- *For any corner tree* $\mathcal{T}$ *and any permutation* Λ *the corner tree count* $\#_{\mathcal{T}}(\Lambda)$ *can be computed in time* $\mathcal{O}(|\Lambda| \log(|\Lambda|))$.

Ways forward:

- Corner tree counting as homomorphism counting in the category of double posets. [2] ✓
- Extend the subspace of patterns covered by effective algorithms. [2] ✓
- Global permutation entropy. [3] ✓
- Use these ideas for other computational problems:
  - chirotope counting (w/ Velankar, next section),
  - iterated sums (last section),
  - others?



[2] J. Diehl and E. Verri. "Efficient counting of permutation patterns via double posets". In: arXiv preprint arXiv:2408.08293 (2024)

[3] A. Avhale, J. Diehl, N. Velankar, and E. Verri. "Global Permutation Entropy". In: arXiv preprint arXiv:2508.19955 (2025)

Overview

Counting permutation patterns

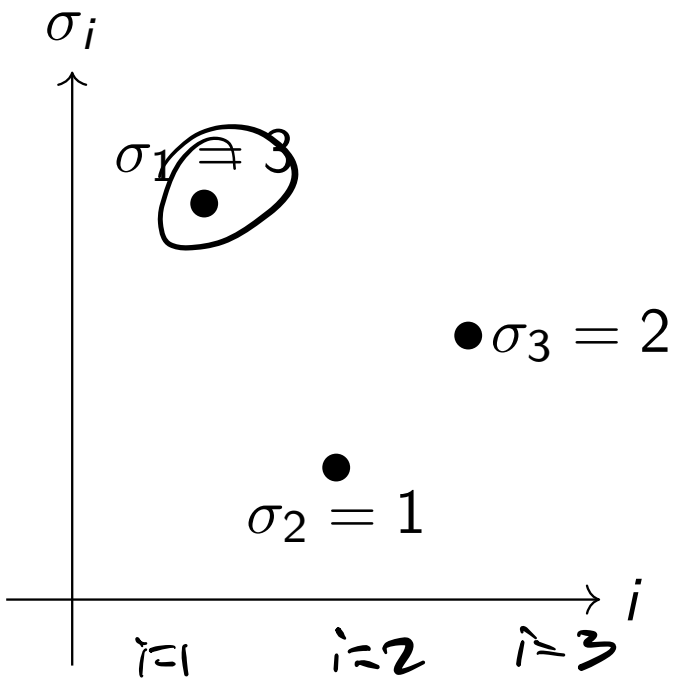
Counting chirotope patterns

Fast iterated sums for tensor-to-tensor layers

From permutations (1D) to point configurations

Revisit 1D (permutations)

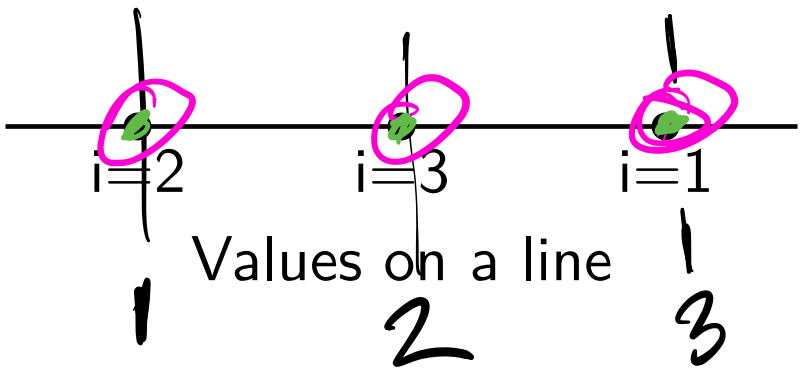
So far: a permutation $\sigma = [\sigma_1, \ldots, \sigma_n]$ has been viewed as points (i, σ_i).



To prepare for 2D, we now consider the "scatterplot".

The "pattern" is determined by relative order. For $i < j$:

$$\operatorname{sgn}(\sigma_j - \sigma_i) = \operatorname{sgn} \det \begin{bmatrix} 1 & \sigma_i \\ 1 & \sigma_j \end{bmatrix}$$

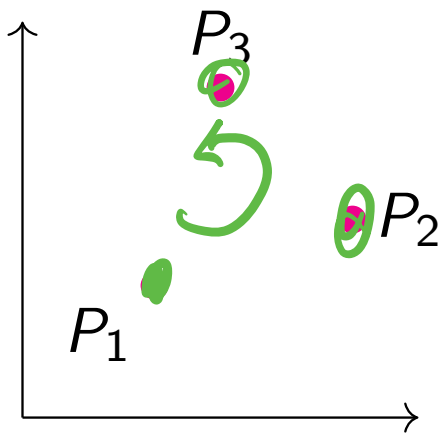


Chirotopes (2D)

Definition: For a set of points $P_1, \ldots, P_n \in \mathbb{R}^2$, the **chirotope** χ records the orientation of every triplet.

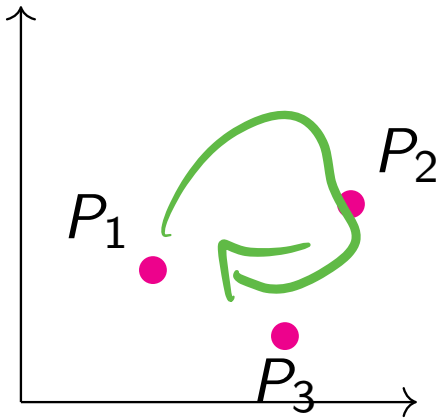
$$\chi(i,j,k) = \operatorname{sgn} \det \begin{bmatrix} 1 & P_i \\ 1 & P_j \\ 1 & P_k \end{bmatrix} \in \{+,-\}$$

(Assuming non-degenerate, generic positions).



Counter-clockwise

$\det > 0 \implies +$

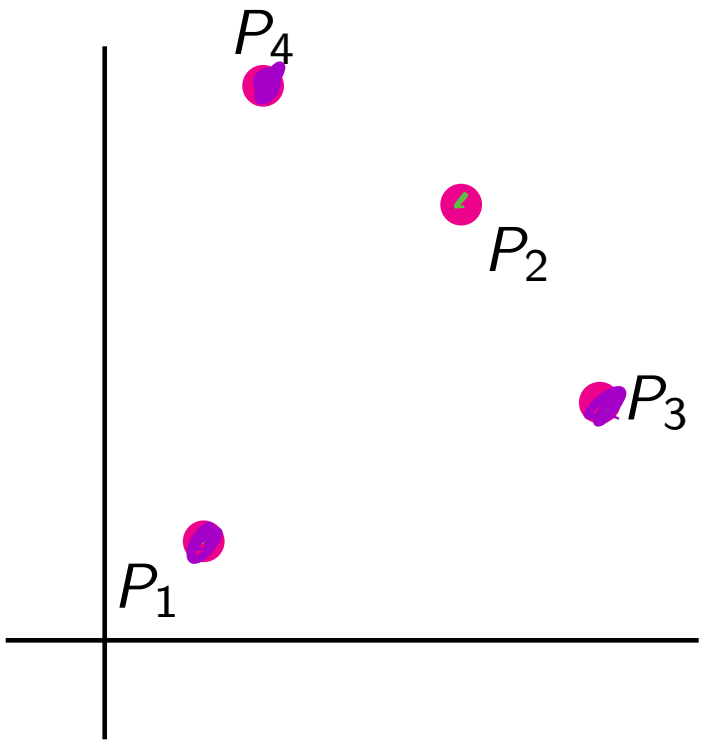


Clockwise

$\det < 0 \implies -$

Example Calculation

Consider a configuration of 4 points. The chirotope is the sequence of signs for all triplets $\binom{4}{3}$.



$$\chi(1,2,3) = - \quad \text{(Clockwise)}$$
$$\chi(1,2,4) = + \quad \text{(CCW)}$$
$$\chi(1,3,4) = + \quad \text{(CCW)}$$
$$\chi(2,3,4) = - \quad \text{(Clockwise)}$$

Signature: $(-++-)$

Analogous to permutation patterns one can define

$$\#_\sigma(\chi) := \#\{\text{“copies” of chirotope } \sigma \text{ in chirotope } \chi\}.$$

Moreover, a two-dimensional time-series X canonically corresponds to a chirotope χ_X (this is actually how we argued on the previous slides ..).

Aside:

$$\#_{(+)}(\chi_X) - \#_{(-)}(\chi_X)$$

can be considered a quantized version of signed area.

Chirotope entropy[4]

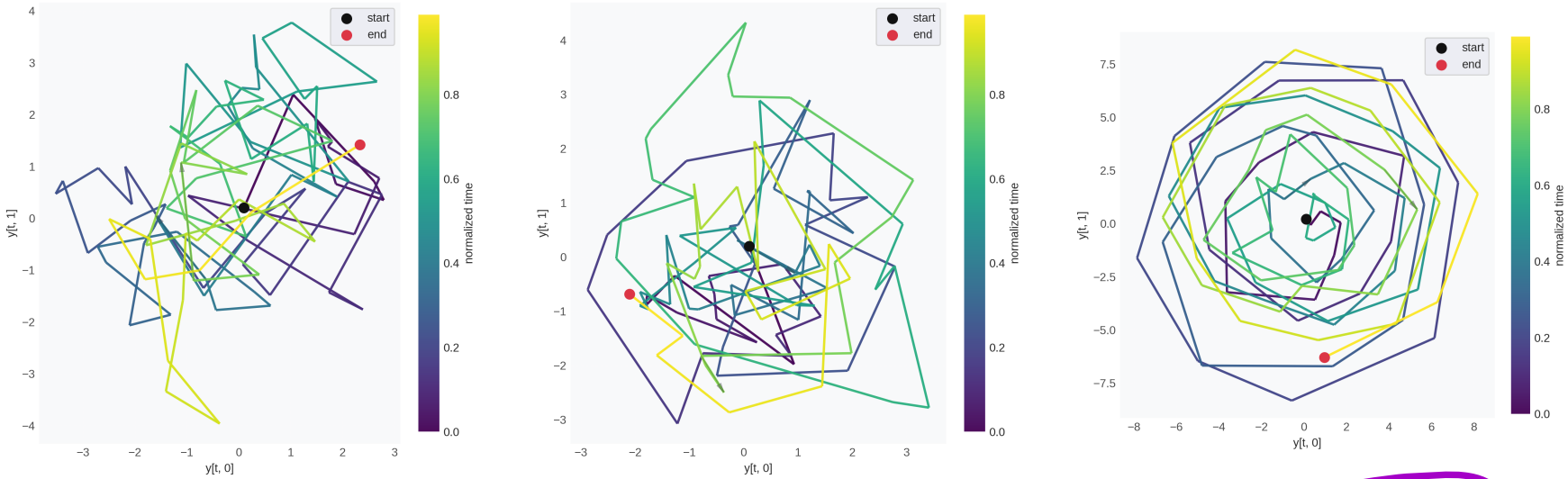
Recipe: count chirotope patterns (of a certain size), and calculate the Shannon entropy of the resulting empirical distribution. [5]

[4] Work in progress with Ch. Bandt, P. Geiger, E. Verri

[5] Currently: consecutive patterns.

Example 1: (Discretized) Brownian motion in a magnetic field.

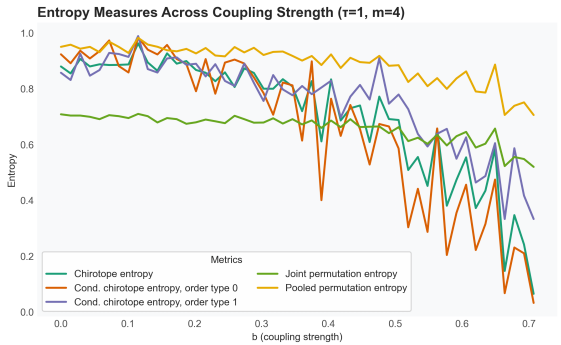
$$y_t = \begin{bmatrix} \frac{1}{\sqrt{2}} & -b \\ b & \frac{1}{\sqrt{2}} \end{bmatrix} y_{t-1} + \epsilon_t, \qquad y_0 \in \mathbb{R}^2.$$



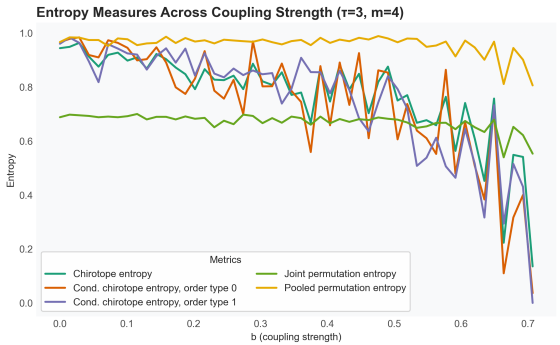
(a) $b = 0.$ (b) $b = 0.375$ (c) $b = 0.7$

Fig: Phase space plots of y for different values of the coupling constant b.

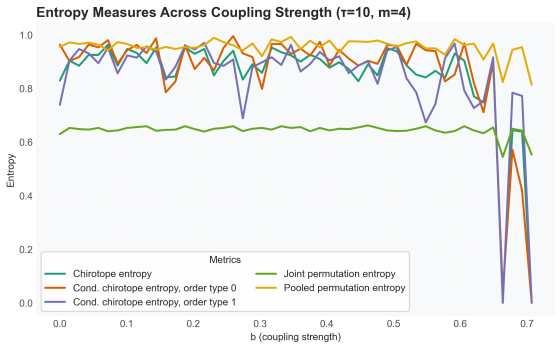
We vary the coupling constant b over the range $[0, \frac{1}{\sqrt{2}}]$, the embedding delay τ over $\{1, 3, 10\}$ and the embedding dimension m over $\{4, 5\}$.



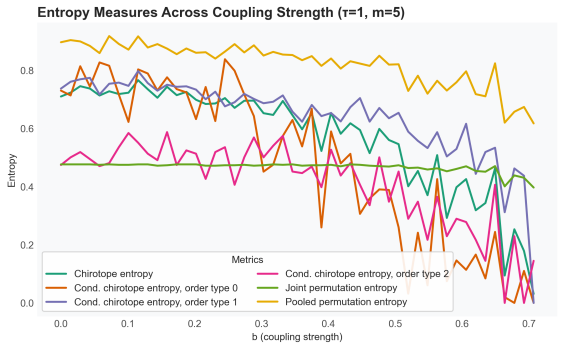
(a) $\tau = 1, m = 4$



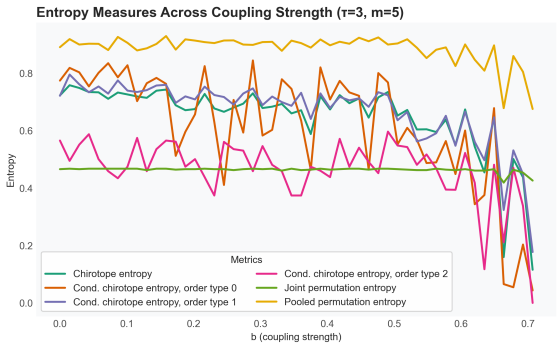
(b) $\tau = 3, m = 4$



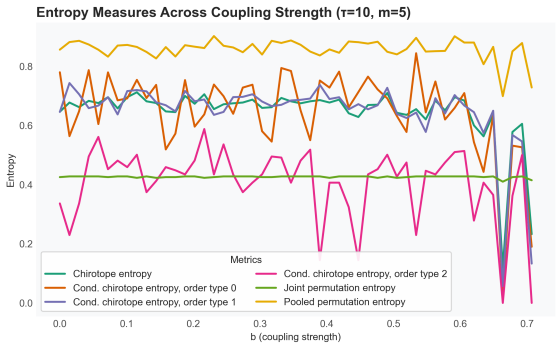
(c) $\tau = 10, m = 4$



(d) $\tau = 1, m = 5$



(e) $\tau = 3, m = 5$



(f) $\tau = 10, m = 5$

Fig: Various entropies for the two-dimensional AR(1) model with different embedding delays τ.

Example 2: Machine failure

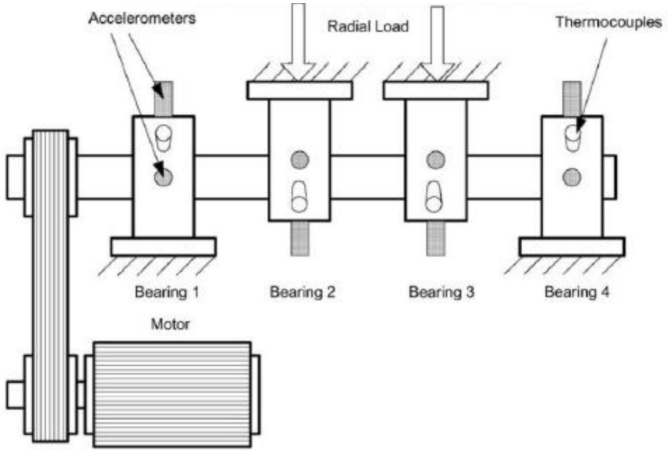
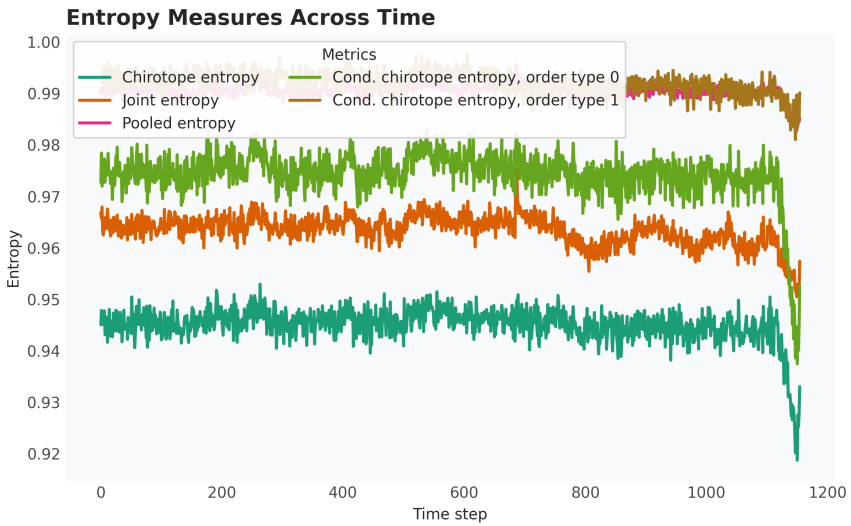
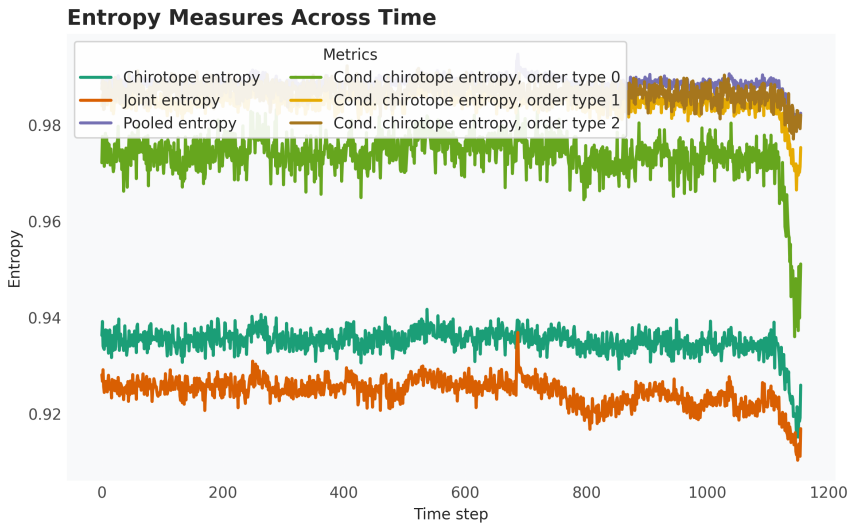


Fig: Bearing with radial load.



(a) $\tau = 5$, $m = 4$



(b) $\tau = 5$, $m = 5$

Fig: Entropy measures for the NASA Bearing Dataset Number 1 for different delays τ

Ways forward:

- counting chirotope patterns (work in progress with N. Velankar); right now: can count 12 dimensions out of 14 for size-4 patterns in subcubic time.
- global chirotope entropy (TBD)
- use algorithmic ideas gained here for ML (TBD)

Overview

Counting permutation patterns

Counting chirotope patterns

Fast iterated sums for tensor-to-tensor layers

Sequence-to-sequence layers are the workhorse of deep learning for sequential data (for example: LLMs).

Successful examples are:

- self-attention / transformer (GPT, BERT, ...),
- infinite-horizon convolutions (Hyena, ...),
- affine state-space models (S4, Mamba, ...)
  $\rightsquigarrow$ iterated sums (Liquid state space models, Elissabeth, ...).

What about image-to-image or tensor-to-tensor?

Sequence-to-sequence layers are the workhorse of deep learning for sequential data (for example: LLMs).

Successful examples are:

- self-attention / transformer (GPT, BERT, ...),
- infinite-horizon convolutions (Hyena, ...),
- affine state-space models (S4, Mamba, ...)
  $\rightsquigarrow$ iterated sums (Liquid state space models, Elissabeth, ...).

What about image-to-image or tensor-to-tensor?

iterated sums look like

$$y_t = \sum_{1 \le i_1 < \cdots < i_k \le t} f_1^\theta(x_{i_1}) \dots f_k^\theta(x_{i_k}), \quad t = 1, \dots, T.$$

- computable in $\mathcal{O}(T)$ and parallelizable,
- in the algebraic setting ($f_i^\theta(x) = x^{\alpha_i}$): quasisymmetric functions,
- can be thought of as discretized iterated integrals (but are in fact much more expressive)
- can be thought of as a non-consecutive convolution layer (but are non-linear)
- allow to switch the semiring, e.g.

$$y_t = \max_{1 \le i_1 < \cdots < i_k \le t} \left(f_1^\theta(x_{i_1}) + \cdots + f_k^\theta(x_{i_k}) \right), \quad t = 1, \dots, T.$$

The animation showed: it is a sum over point configurations. For example,

$$\sum_{1 \le t_1 < t_2 \le t} x_{t_1} x_{t_2}^2,$$

sums over all pairs of integer points (t_1, t_2) in the interval $[1, T]$ which satisfy $t_1 < t_2$:

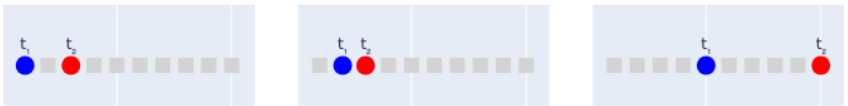


Fig: Example point constellations for the iterated sum $\sum_{1 \le t_1 < t_2 \le T-1} x_{t_1} x_{t_2}^2$.

This viewpoint can be generalized to two-parameter data, as is done in[6] (see[7] for the continuous case) for arbitrary "point constellations" in the plane. For example, the quadruples of allowed points visualized here and:

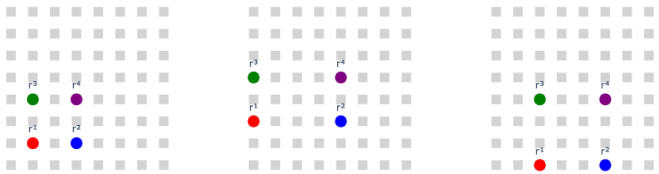


Fig: Example point constellations for the sum $(*)$.

lead to a sum of the form

$$\sum_{\mathbf{r}^1,\mathbf{r}^2,\mathbf{r}^3,\mathbf{r}^4 \text{ allowed}} z_{\mathbf{r}^1} z_{\mathbf{r}^2} z_{\mathbf{r}^3} z_{\mathbf{r}^4} . \qquad (*)$$

Such expressions are sufficiently expressive (they characterize the input, up to a natural equivalence relation), but are, in general, not computable in linear time.

[6] J. Diehl and L. Schmitz. "Two-parameter sums signatures and corresponding quasisymmetric functions". In: arXiv:2210.14247 (2022).

[7] J. Diehl, K. Ebrahimi-Fard, F. N. Harang, and S. Tindel. "On the signature of an image". In: Stochastic Processes and their Applications (2025), p. 104661; C. Giusti, D. Lee, V. Nanda, and H. Oberhauser. "A Topological Approach to Mapping Space Signatures". In: arXiv preprint arXiv:2202.00491 (2022).

($z : [T_1] \times [T_2] \to \mathbb{R}^d$ an "image" that we want to process.)

Corner trees for fast iterated sums

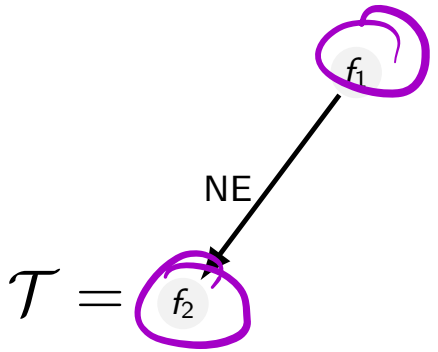
Let $\mathcal{C} := \{\mathrm{N}, \mathrm{NE}, \mathrm{E}, \mathrm{SE}, \mathrm{S}, \mathrm{SW}, \mathrm{W}, \mathrm{NW}\}$.

corner tree:

$\mathcal{T} = (V(\mathcal{T}), E(\mathcal{T}), \mathfrak{v} : V(\mathcal{T}) \to \mathrm{Map}(\mathbb{R}^d, \mathbb{R}), \mathfrak{e} : E(\mathcal{T}) \to \mathcal{C})$

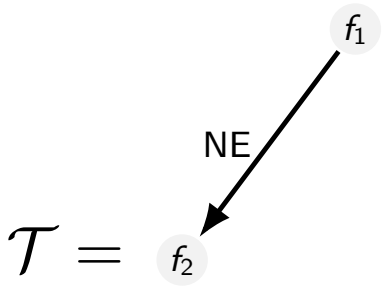
corresponding iterated sum:

$$\sum_{\substack{\mathbf{r}: V(\mathcal{T}) \to [T_1] \times [T_2] \\ \text{"compatible with } \mathcal{T}\text{"}}} \prod_{v \in V(\mathcal{T})} \mathfrak{v}(v)(z_{\mathbf{r}(v)}).$$



$$\sum_{\substack{\mathbf{r}^1, \mathbf{r}^2 \in [0, T_1] \times [0, T_2] \\ (r_1^1 < r_1^2) \wedge (r_2^1 < r_2^2)}} f_1(z_{\mathbf{r}^1}) f_2(z_{\mathbf{r}^2}).$$

Visualization for the example:



$$\sum_{\substack{\mathbf{r}^1,\mathbf{r}^2\in[0,T_1]\times[0,T_2]\\(r_1^1<r_1^2)\wedge(r_2^1<r_2^2)}} f_1(z_{\mathbf{r}^1})f_2(z_{\mathbf{r}^2}).$$

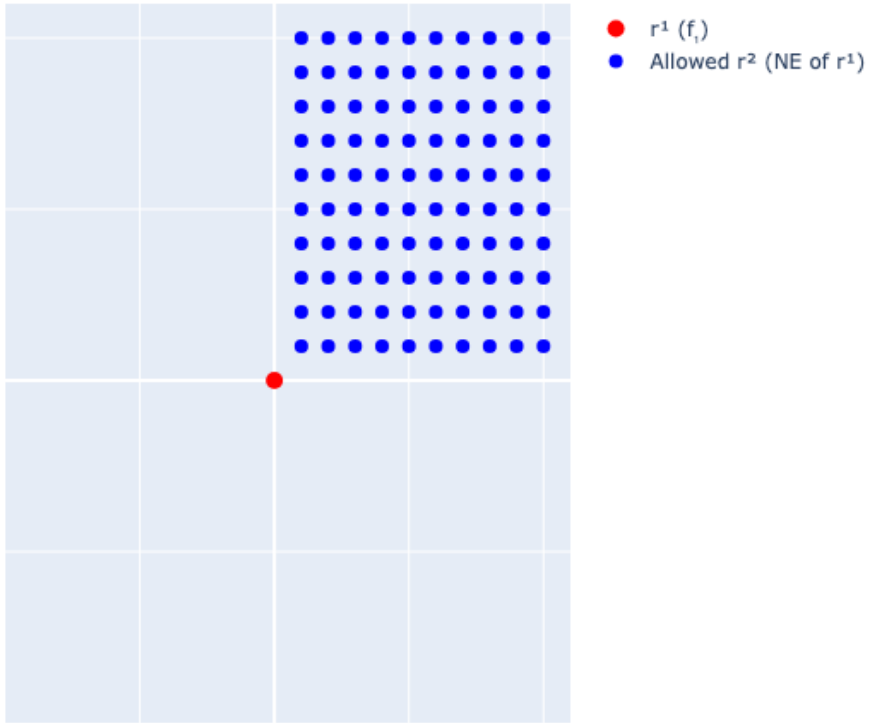
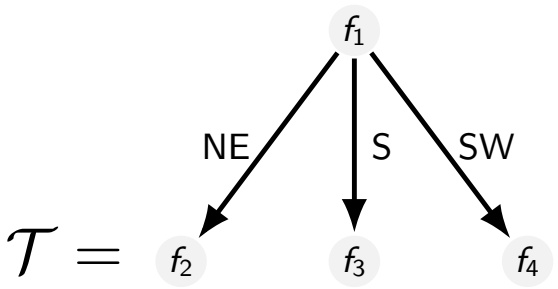


Fig: Point constellations for the corner tree $\mathcal{T}$.

Two more examples:



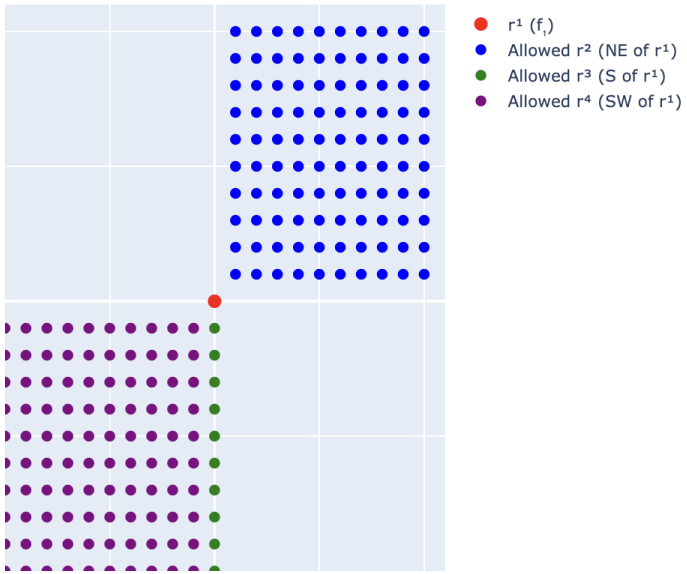
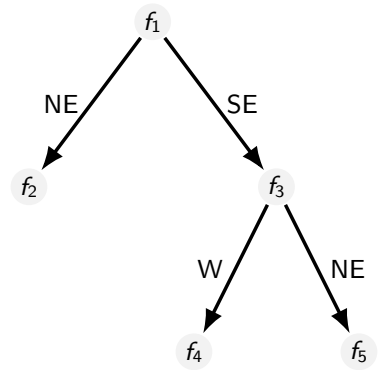


Fig: Point constellations for the corner tree $\mathcal{T}$.

The trees can have arbitrary depth:

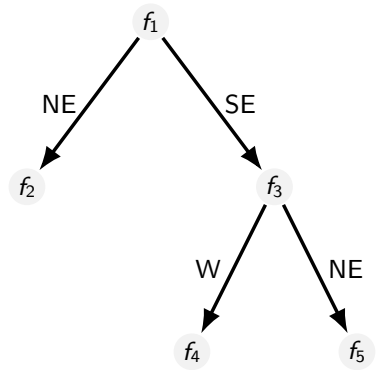
$\mathcal{T} =$ 

Theorem (D, Ibraheem, Schmitz, Wue[8])

For any corner tree $\mathcal{T}$ and any image $z : [T_1] \times [T_2] \to \mathbb{R}^d$ the corner tree iterated sum for $\mathcal{T}$ in z can be computed in time $\mathcal{O}(T_1 T_2)$.

[8] J. Diehl, R. Ibraheem, L. Schmitz, and Y. Wu. "Tensor-to-Tensor Models with Fast Iterated Sum Features". In: Neurocomputing (2026).

The trees can have arbitrary depth:

$\mathcal{T} =$ 

Theorem (D, Ibraheem, Schmitz, Wue[8])

For any corner tree $\mathcal{T}$ and any image $z : [T_1] \times [T_2] \to \mathbb{R}^d$ the corner tree iterated sum for $\mathcal{T}$ in z can be computed in time $\mathcal{O}(T_1 T_2)$.

[8] J. Diehl, R. Ibraheem, L. Schmitz, and Y. Wu. "Tensor-to-Tensor Models with Fast Iterated Sum Features". In: Neurocomputing (2026).

Experiments

We use the FIS layer as a drop-in replacement for certain convolutional layers of `ConvNeXt`.[9]

Fig: The two corner trees used in the FIS drop-in layer in the ConvNeXt experiments.

Model	Variant	Accuracy (%)	Params (M)	time / epoch
ConvNeXt tiny	Base	93.88, 73.70*	27.83	437.66
ConvNeXt tiny	L2	93.77, 74.53*	27.72	401.30
ConvNeXt tiny	L4	94.48, 75.55*	27.17	402.10

Fig: Results on CIFAR-10 and CIFAR-100.

[9] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. "A convnet for the 2020s". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022), pp. 11976–11986.

Ways forward:

- `pytorch` layer. ✓
- Use as drop-in replacement for convolution layers in existing architectures (`ResNet`) shows improvement on classification and anomaly detection tasks. ✓
- Build a deep architecture from scratch using fast iterated sums.

I welcome questions and comments.